Softwarekonstruktion - Exercise 5

5 Design by Contract

This exercise should be solved until **Wednesday (23:59 latest), November 24th, 2010**. You have to submit your solution to your tutor by email:

Holger Schmidt: holger.schmidt [at] cs.tu-dortmund.de
Gregor Kotainy: gregor.kotainy [at] tu-dortmund.de

You have to work in groups of two or three persons. Only one person per group has to submit a group's solution. State the names and matriculation numbers of the group members in your email and as a comment in each of your source code files.

5.1 Pre- & Postconditions

Given are the classes `Element` and `Queue`. A queue consists of elements to which an integer-value has been assigned. An element holds a reference to its successor element.

class	**Element**
attribute	value: Integer
	next_element: Element
method	setNextElement(new_element: Element)
	getNextElement(): Element
	getValue(): Integer
end class	**Element**

class	**Queue**
attribute	nb_elements: Integer
	max_size : Integer
	first_element : Element
	last_element : Element
method	empty(): Boolean
	full(): Boolean
	add(new_element: Element)
	remove(): Element
	peek(): Element
end class	**Element**

Specify pre- and postconditions for the operations of the class **Queue** (similar to the class **Stack** as presented in the lecture notes pp. 165). Note that you have to submit your solution electronically in PDF format. For example, you can use LaTeX, or you can prepare a scan of your handwritten solution.

`empty:` returns `true` if the queue is empty.

`full:` returns `true` if the queue is full.

`add:` adds the specified element to the end of the queue.

`remove:` retrieves and removes the first element of the queue.

`peek:` retrieves, but does not remove, the first element of the queue.

5.2 Java

Implement the classes `Element` and `Queue` in Java. When creating a new *Java Project* in *Eclipse*, choose *Use default JRE*. Use *assertions* (http://java.sun.com/j2se/1.4.2/docs/guide/lang/assert.html) to test the pre- and postconditions of the methods `add`, `remove` and `peek`:

- Use the following form of the assertion statement:
  `assert` $Expression_1$ `:` $Expression_2$
  When the system runs the assertion, it evaluates $Expression_1$ and if it is `false` throws an `AssertionError`. The system passes the value of $Expression_2$ to the `AssertionError` constructor, which uses the string representation of the value as the error's detail message.
- Because assertions may be disabled, programs must not assume that the boolean expression contained in an assertion will be evaluated. Do *not* use assertions to do any work that your application requires for correct operation.
- Occasionally it is necessary to save some data prior to performing a computation in order to check a postcondition. You can do this with two `assert` statements:

```
int oldValue = 0;
assert (oldValue = nb_elements) == nb_elements :  "saving data"
...
assert nb_elements == oldValue + 1 :  "nb_elements not increased";
```

- To enable assertions in Eclipse, go to *Window* → *Preferences* and select *Java* → *Installed JRE's*. Select your JRE, click *Edit* and enter **-ea** under *Default VM Arguments*.